



## **CLOUD SOFTWARE PROGRAM**

### **REPORT**

**WP1: Mashupper – Agent-enabled Social  
Cloud  
Nokia & JYU**

**Q1-Q2/2011**

Michael Cochez  
Michal Nagy

miselico@jyu.fi  
minagy@jyu.fi

## ***Table of Contents***

1	Introduction.....	3
2	Short description of the Ubiware platform .....	4
2.1	Typical Ubiware agent .....	4
2.2	Latest generation of Ubiware platform .....	5
3	Social Ontology .....	7
4	Agent Architecture.....	8
5	Web application user interface.....	10
5.1	Adapter authentication .....	10
5.2	Building a Personal User Network.....	11
5.3	Using Personal User Network .....	13
5.4	Geographical status updates panel .....	14
5.5	Activity timeline.....	15
6	Meego version of Mashupper .....	16
7	Conclusions.....	17

# 1 Introduction

The Mashupper application uses data from three prominent social network platforms: Facebook, LinkedIn and Twitter. As the largest player in the field with more than 500 million users, Facebook is becoming a virtual world of its own. It fulfills the need for general socializing in the web and covers both leisure and work. LinkedIn on the other hand is very much profiled to the business and professional side of the social networking. Then there is the Twitter, which has capitalized on the people's need to hear and to be heard, preferably in real-time. The social connections in Twitter are looser than in LinkedIn or Facebook. Everything you write to Twitter (or tweet) is public and can be read by anyone. In Twitter, user can start to follow his or her friends or actually anyone who seems interesting enough, in order to receive updates from those people in real-time.

Facebook, LinkedIn and Twitter are a good set of services to start with, but there are many other interesting social networking sites out there. Adding new information sources to Mashupper is relatively simple task, thanks to the agent-based architecture of UBIWARE platform, given of course that the social network service provides some kind of API for external services.

The scope of the Mashupper is built on top of the concept of Personal User Network (PUN), which is defined as the combination of different kinds of human connections, which a particular user may have in the social web.

PUN is used to link the different online identities or profiles into one and same connection, through which the user can observe the integrated presence of that connection in the current Web2.0 landscape.

There are two versions of Mashupper application – a web-based application and a Meego mobile application. The web application can be used both for contact matching and viewing PUN. The Meego version can be used only for viewing PUN. The reason for this is that contact matching is not a very common operation and it would perhaps become too complicated for a mobile interface. Also, it does not increase the scientific value of this research.

Mashupper web application was developed using web development framework called Vaadin<sup>1</sup>. Vaadin applications are written in Java programming language and the development kit already contains a collection of UI components. Since no knowledge of HTML, CSS or Javascript is needed, this approach is suitable for developers used to Java Swing user interfaces.

The Meego mobile version of the interface was developed using QML (Qt Modeling Language) language. It is a new declarative language build on top of Qt. This language eases the development of fluid animations and touch input, while still leaving the freedom to write native code on the mobile platform.

Before going into details of Mashupper application, we will shortly discuss the Ubiware platform on which Mashupper was developed.

---

<sup>1</sup> <http://vaadin.com/>

## 2 Short description of the Ubiware platform

Ubiware was a TEKES-funded project that the Industrial Ontologies Group (IOG) lead between years 2007 and 2010. During the project an agent platform with the same name was developed. Ubiware platform is a middleware agent platform that allows creation of self-managed complex systems by interconnecting components of different nature, e.g. software components, hardware component, humans, etc. For the rest of the document the word Ubiware refers to the platform, unless explicitly specified otherwise.

Currently Ubiware acts as a layer on top of JADE agent system<sup>2</sup>. Each Ubiware agent is programmed using so-called Semantic Agent Programming Language (S-APL). S-APL is a declarative rule-based language that is based on Resource Description Framework (RDF) and Notation 3. In comparison to RDF, S-APL adds new concept of container. Container represents a reification. This way it is possible to make statements about a set of statements. An example of S-APL can be seen in Figure 1.

```
//Printing Hello World.
{
  sapl:I sapl:haveName ?myName
} => {
  {sapl:I sapl:do :Print} sapl:configuredAs {
    p:print sapl:is "Hello World! My name is ?myName"
  }
}.

// This starts the agent web server on port 8181
{sapl:I sapl:do
java:ubiware.shared.agentServer.AgentServerBehavior}
sapl:configuredAs {
  p:port sapl:is "8181" .
} .
```

Figure 1 – Example of S-APL

### 2.1 Typical Ubiware agent

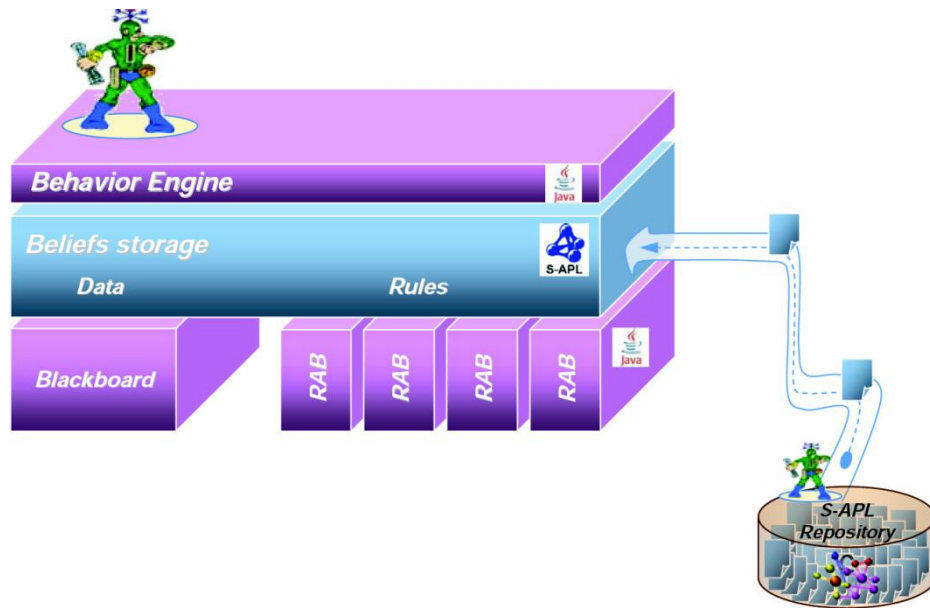
A Ubiware agent consists of three main layers – behavior engine, belief storage and Reusable Atomic Behaviors (RABs). The schema of a typical agent can be seen in Figure 2.

RAB is a Java class implementing a reasonably atomic function. RABs are called reusable, because they can be reused across scenarios, agents and roles. The agent programmer can call a RAB with some given parameters and it will execute the task it was designed for. A few examples of RAB functions are sending of messages, receiving of messages, accessing a database, downloading a resource, etc.

The belief storage is a database that contains all the beliefs of the agent. These beliefs can be just pure beliefs like “John loves Mary” or they can be rules. In Ubiware rules are considered just special beliefs. Thanks to this principle you can create new rules or change existing ones. A rule consists of left side (query) and the right side (effect). Whenever the left side is true, the right side will be added to the belief storage.

---

<sup>2</sup> <http://jade.tilab.com/>



**Figure 2 – Three-layered hierarchical model of a Ubiware agent**

Each agent has its own copy of the behavior engine, therefore they are independent. The engine is executing so-called Live behavior. Live behavior is cycling through the agent beliefs and it is looking for rules or RAB calls that can be executed. This behavior constantly updates the beliefs of the agent.

## **2.2 Latest generation of Ubiware platform**

The Ubiware platform has gone through several years of development. In its latest stage we loosened the ties with JADE platform and instead of using some of JADE's original functionality we added some of our own. Also, the platform started to resemble an operating system, where various users can use various applications. Currently there are several types of agents – platform infrastructure agents, application infrastructure agents, personal user agent and personal worker agent.

There are exactly seven platform infrastructure agents on every Ubiware platform. They exist on the platform from its creation till its termination. They perform generic platform-wide services to other agents and application. An example would be the Web Interface Agent providing HTTP access to the platform.

Application infrastructure agents are unique for each application and as the name suggests they provide application-wide functionality.

Personal user agent is representing the human in the platform. Its main role is to create and manage personal work agents whenever the user wants to access a certain application.

Whenever Ubiware platform is started, it already contains seven platform infrastructure agents. It also may include several other agents based on how many applications and users are on the platform. In general the platform is dynamic and the number of agent varies over the time. The platform architecture can be seen in Figure 3

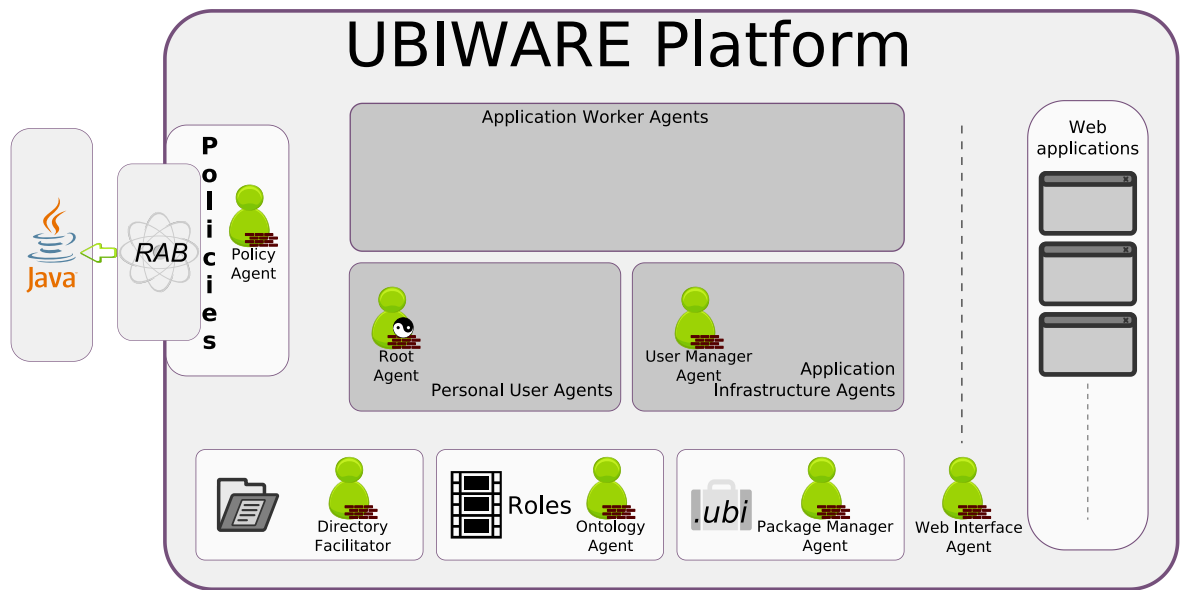


Figure 3 – Ubiware platform architectural overview

### 3 Social Ontology

Social ontology is a domain specific simple ontology for the social web. The main concepts and their properties are depicted in the Figure 4. Personal user network (PersonalUserNetwork class) can be thought of as set of people that the user has some kind of connection with. These connections are modeled using HumanConnection class. The name was chosen to emphasize the fact that connection refers to a single human being. This person can then be represented in the social web by multiple digital identities, modeled in the ontology as SocialNetworkProfiles. SocialNetworkProfile is the common parent of more service-specific profiles such as FacebookProfile, LinkedInProfile and TwitterProfile.

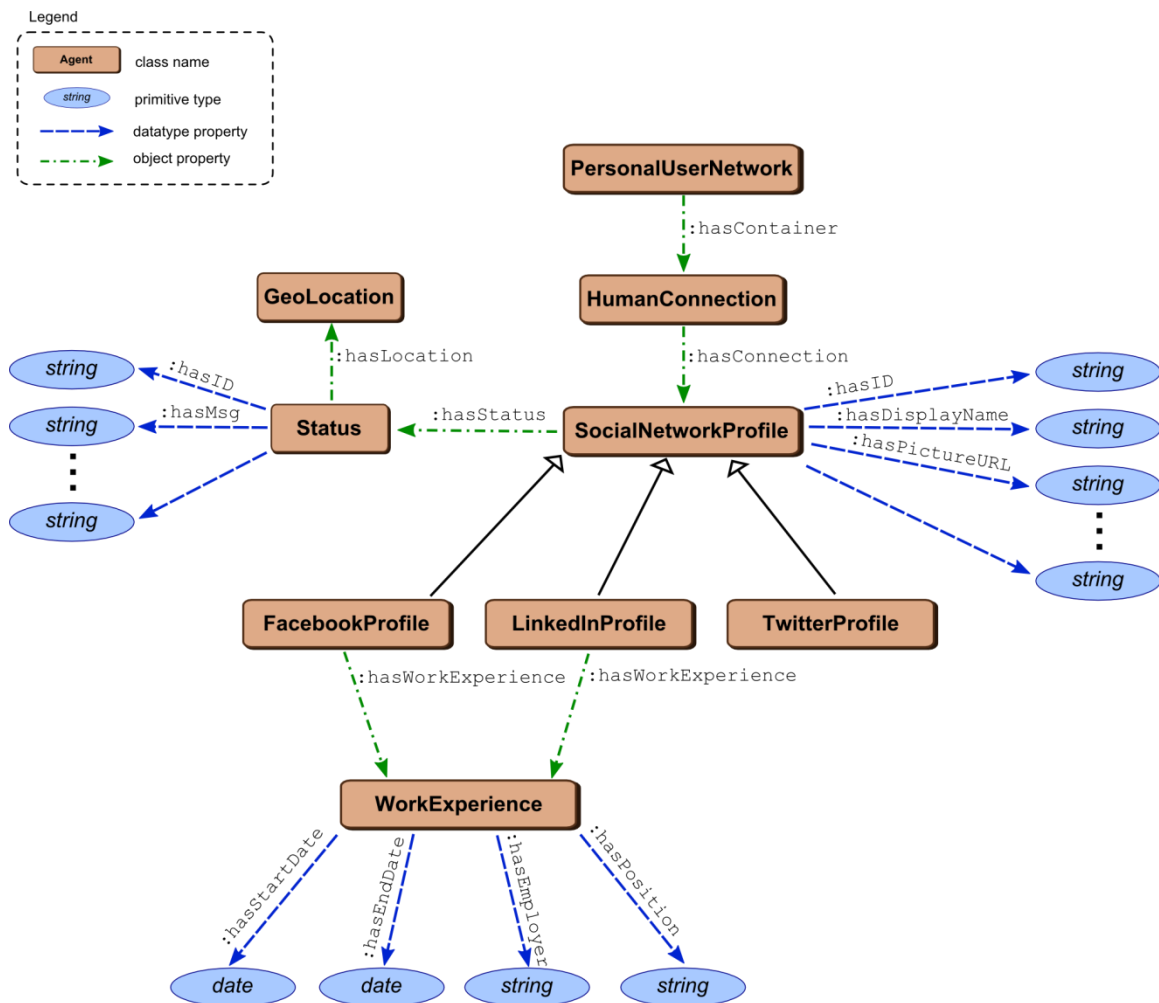


Figure 4 – Social Ontology

The concept Status refers to any kind of state information available. It is used to bring together status updates from Facebook, network updates from LinkedIn and tweets from Twitter. After all, they are all used to represent the current state, or status of mind of the particular HumanConnection at a particular moment in time. Location, as an increasing important factor of the social web, can also be included as part of the Status using GeoLocation class.

## 4 Agent Architecture

The Mashupper consists for the user of two applications. In addition to the web application providing main Mashupper UI, there is one web application for authenticating external sources for the ubiware platform. Behind the scenes, there is an invisible application for each available social network adapter.

The Mashupper is designed to run on the Ubiware desktop. When using the mashupper application, the user gets a worker agent for each application in use. This means that there is one for the actual Mashupper, one for the Authenticator and then one for each adapter selected by the user. Normal infrastructure agents are naturally also active. Figure 5 shows a possible setup of agents involved and their respective roles. (In this scenario, Facebook and LinkedIn are selected by the user.)

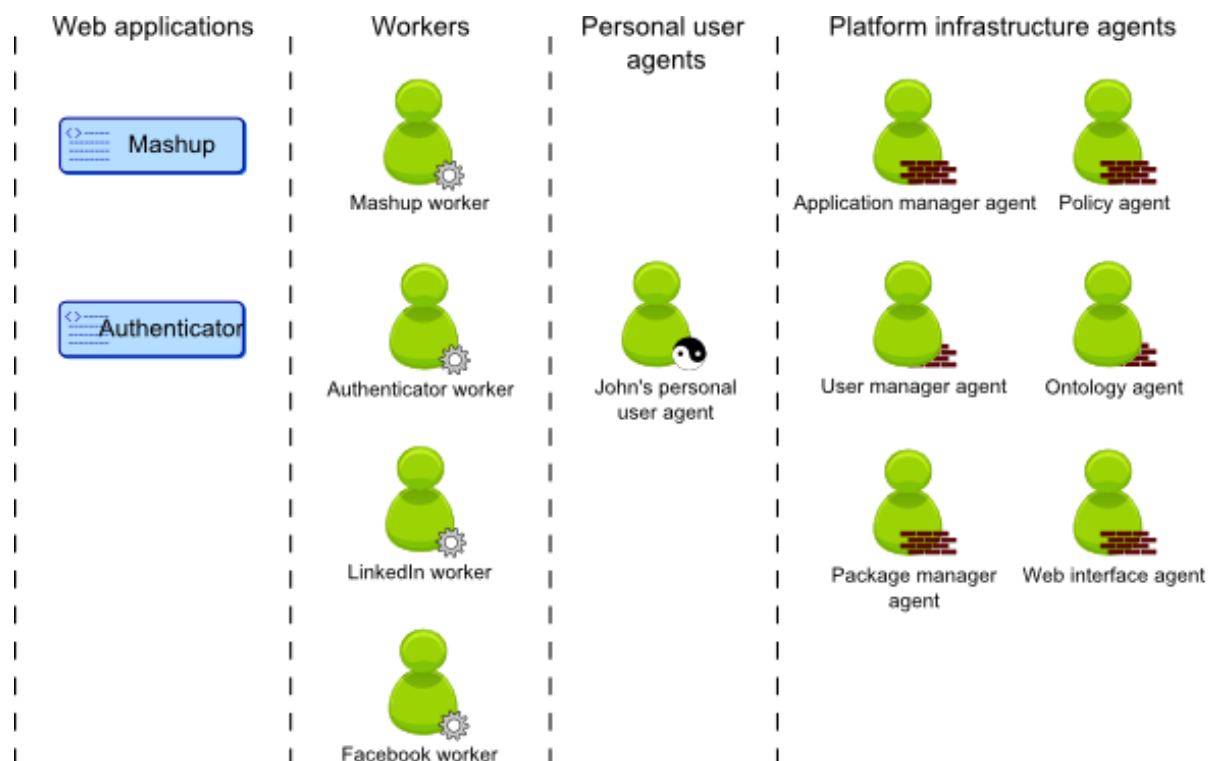


Figure 5 – Mashupper Application Agents

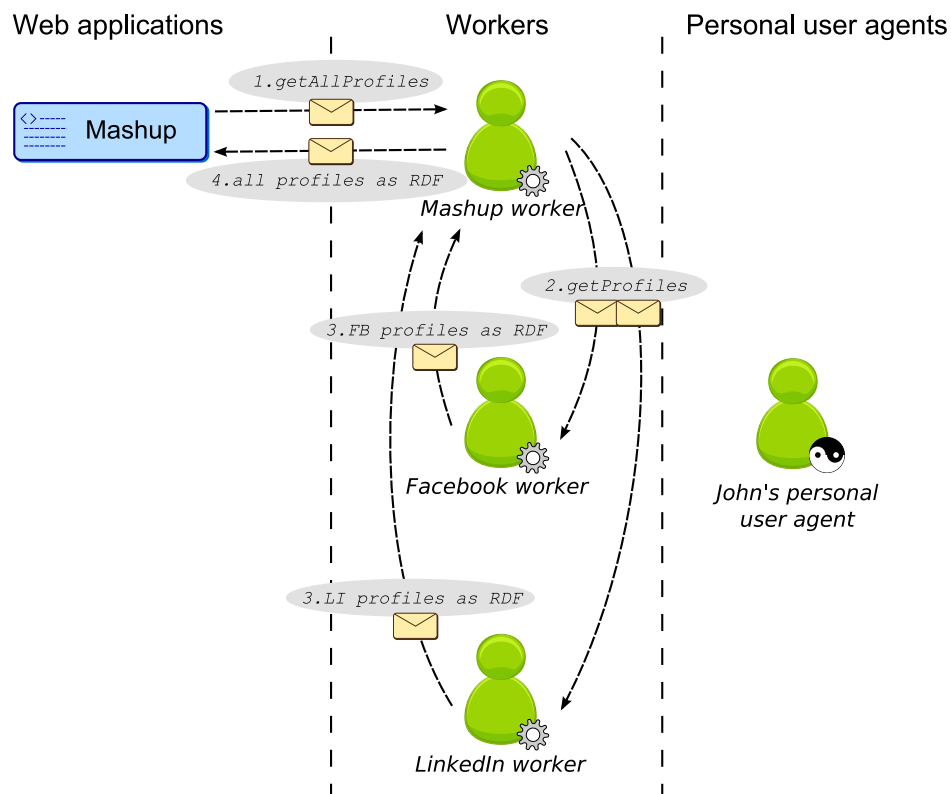
MashupWorker is responsible for storing the knowledge about the Personal User Network of the user and works directly with Mashupper web application. The Authenticator worker is responsible for providing the user a homogeneous way to authentication of various external resources (In this case social networks). Adapter workers, like for instance Facebook worker, handle all the information related to the external service API, such as URLs of the service interface, client ID, keys for signing and access tokens for authorizing requests. They also know how to access the external service using API-specific RABs. There is one worker for Facebook, LinkedIn and Twitter.

Figure 6 shows an example of agent interaction coordinated by the MashupWorker, where web application requests for detailed profile information for a certain HumanConnection. In this case the HumanConnection has been linked to both Facebook and LinkedIn social network profiles. The MashupWorker queries worker agents of both services,



waits for their results and returns the combined information to the web application in RDF format.

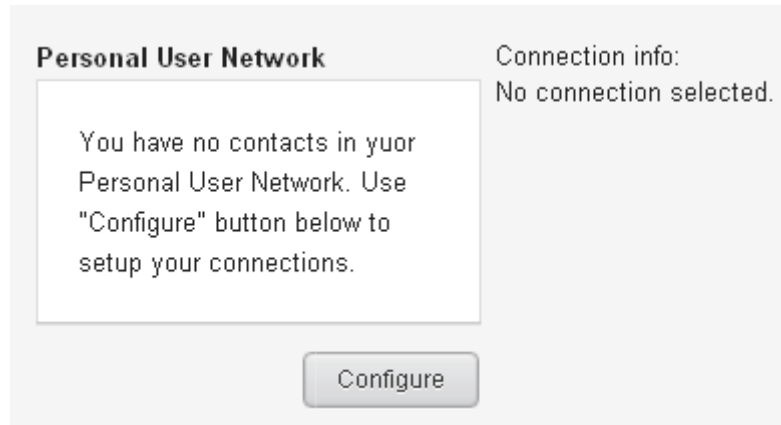
In the current use case Mashupper is the only agent using the information delivering services provided by social network adapters. Nothing stops other applications in the platform from taking the advantage of the same information for the same user. Or even better, they can use the Personal User Network stored at the Mashupper agent and target their actions to a certain HumanConnection instead of a single service. In other words, Mashupper can be reused as a component in future application scenarios.



**Figure 6 – Mashupper agent interaction**

## 5 Web application user interface

When a user starts the web version of Mashupper for the first time, the PUN is naturally empty as shown in Figure 7.



**Figure 7 – Personal User Network is initially empty**

User should initiate the process of Personal User Network construction manually by starting the configuration, which can be accessed by clicking “Configure” button. Configuration view shows the list of available social network adapters and the content of the users of current Personal User Network. Initially, all the adapters are in the offline or non-authenticated state. User must activate adapters by granting the access to its accounts in social networks as explained in the next section.

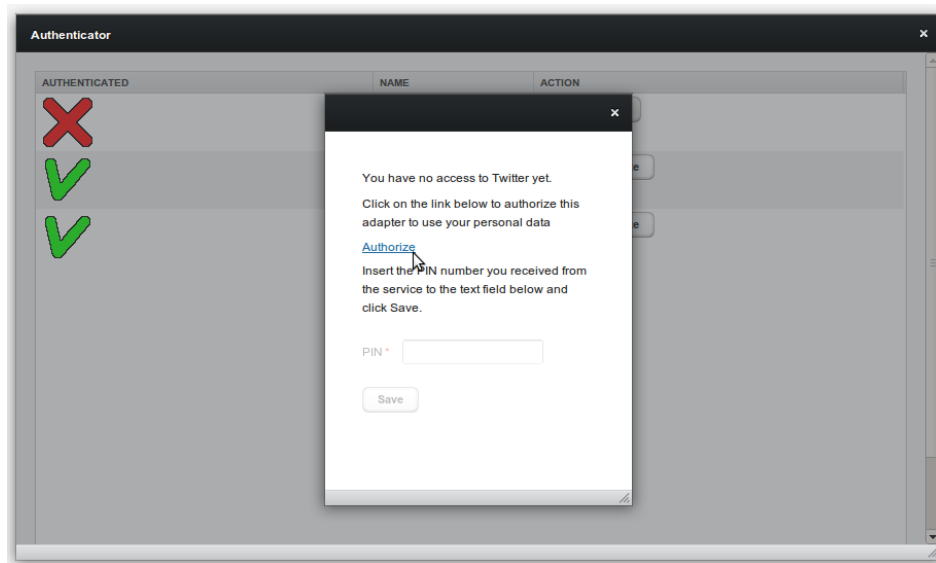
### 5.1 Adapter authentication

All three before mentioned social networks use the Open Authentication<sup>3</sup> (OAuth) method of user authentication. The user has to give the permission to an application to use his/her data. Since authentication is a common paradigm when using external services, we developed an independent authentication application called Authenticator.

Whenever the user wants to give an application on the Ubiware platform the permission to use some external resource he or she owns, the user uses Authenticator to authenticate the platform to the external resource. After the permission is given, any Ubiware application started by the same user can potentially have access to his/her social network information. A social network adapter can be authenticated by clicking on the activate button next to the authenticable external resource. This opens up the UI of the adapter with the instructions how to authenticate and authorize the selected adapter. Depending on the authentication mechanism in use, the interface will be different. An example for authentication with Twitter can be seen in Figure 8.

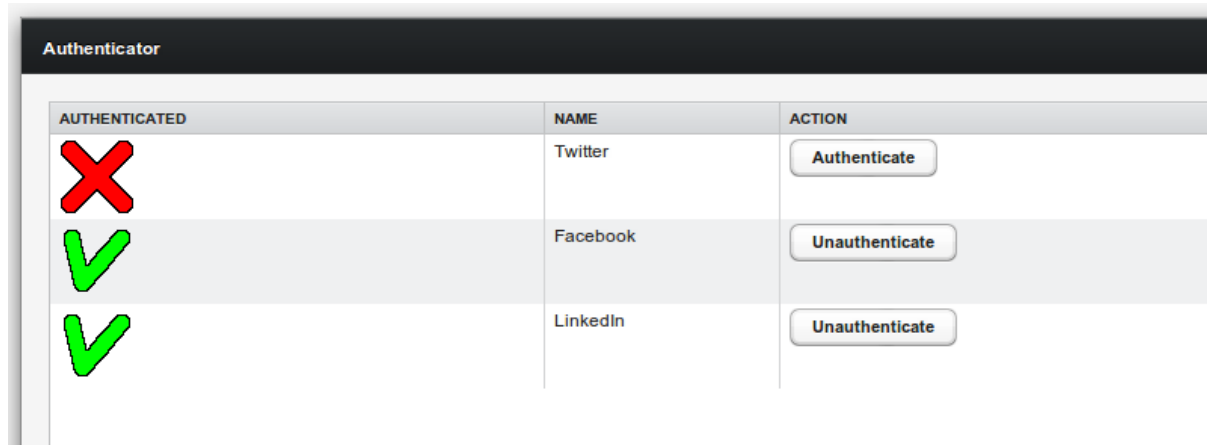
---

<sup>3</sup> <http://oauth.net/>



**Figure 8 – Activation UI for adapters**

In the Twitter example, after following the “Authorize” link and saving the PIN code provided by the external service, the social network adapter UI displays the updated state of the adapter to the user. The Authenticator’s list of adapters now shows the LinkedIn adapter as authenticated and ready for use (Figure 9). The authentication process should be completed once for all the adapters from which the user wants to retrieve data.



**Figure 9 – The list of social network adapters refreshed**

Adapters remain active and authenticated until the user manually revokes the access token it has received in the authentication process.

## **5.2 Building a Personal User Network**

When the appropriate adapters have been authenticated the user can start building his or hers Personal User Network. The user can retrieve the list of people with whom he or she has any kind of contact using the “Update” link associated with every adapter (Figure 10). In Facebook this means the list of friends, in LinkedIn, - the connection or contacts of the user and in Twitter – the list of people the user is following (Figure 11).


Social network adapters			Personal User Network	
NAME	AUTHENTICATED	ACTION	f FACEBOOK	in LINKEDIN
Facebook	true	<a href="#">Update</a> 		
LinkedIn	true	<a href="#">Update</a>		
Twitter	true	<a href="#">Update</a>		

Figure 10 – Updating Personal User Network with friends from Facebook.

Social network adapters			Personal User Network	
NAME	AUTHENTICATED	ACTION	f FACEBOOK	in LINKEDIN
Facebook	true	<a href="#">Update</a>	Pete Smith	empty
LinkedIn	true	<a href="#">Update</a>	Anne Walters	empty
Twitter	true	<a href="#">Update</a>		

Figure 11 – After a successful import, the connections are added to Facebook column of the Personal User Network.

Figure 12 shows the state of PUN after user has retrieved contacts from all the available adapters. Each row in the PUN table represents one human connection. Mashupper supports simple merging of the data using the display name. In the example it has automatically merged Pete Smith from Facebook and LinkedIn into the same connection.


Social network adapters			Personal User Network		
NAME	AUTHENTICATED	ACTION	f FACEBOOK	in LINKEDIN	🐦 TWITTER
Facebook	true	<a href="#">Update</a>	empty	empty	pesmith
LinkedIn	true	<a href="#">Update</a>	Anne Walters	empty	empty
Twitter	true	<a href="#">Update</a>	empty	empty	Katie Woods
			Pete Smith	Pete Smith	empty
			empty	Jack Simpson	empty

Figure 12 – State of the PUN after all adapter updates

The same Pete Smith is also found in the Twitter column, but under the name “pesmith”. The user can now manually link “pesmith” in Twitter with the “Pete Smith” in

Facebook and LinkedIn by drag-and-dropping the name “pesmith” onto to the same row with two other Petes. Mashupper will replace the “empty” label with “pesmith” and remove the first, now empty, connection (Figure 13).

After user has created all the necessary links between different online identities, the PUN is saved to the MashupperWorker agent by clicking the “Save” button.

Personal User Network		
 FACEBOOK	 LINKEDIN	 TWITTER
Anne Walters	empty	empty
empty	empty	Katie Woods
Pete Smith	Pete Smith	pesmith
empty	Jack Simpson	empty

**Figure 13 – Organizing PUN by linking profiles from different networks**

### 5.3 Using Personal User Network

The so far described functionality can only be used through the web version of Mashupper and is not available in the Meego version. However, from this point on, the described functionality is also available through the mobile version. For more information about mobile version user interface, refer to section 6.

When the user opens up the web version of Mashupper application, the content of his or her Personal User Network is shown in the left column. The table lists the names of HumanConnections in the Personal User Network (Figure 14). Each connection is linked to one or more social network profiles as described in the previous section. When the user selects one of the connections, Mashupper uses the associated profiles to retrieve information about that person from the external services. In the current implementation, information sources are Facebook, Linked and Twitter. The retrieved information is shown using three different panels.

Personal User Network	
CONTACT ▲	
Anne Walters	
Jack Simpson	
Katie Woods	
Pete Smith	

**Figure 14 – PUN lists the human connections**

### Profile panel

The profile panel shows the combined profile information about the selected person (Figure 15). There are also links to the person's profile page in all the "adapted" services. The current version of the profile panel includes only basic personal fields. Work experience is retrieved from both Facebook and LinkedIn and combined into single list. If the user hovers the mouse pointer over the values in the table, Mashupper shows the source of that particular piece of information. For example in Figure 16 the name "Pete Smith" was found in both Facebook and Linked and merged as single value, but in Twitter the person uses alias "pesmith".



Figure 15 – A profile panel view



Figure 16 – The source of the information is shown by pointing the mouse over it

## 5.4 Geographical status updates panel

The next panel uses Google Maps to display geo-tagged status updates on a map (Figure 17). The user can click on the marker to display the status message.

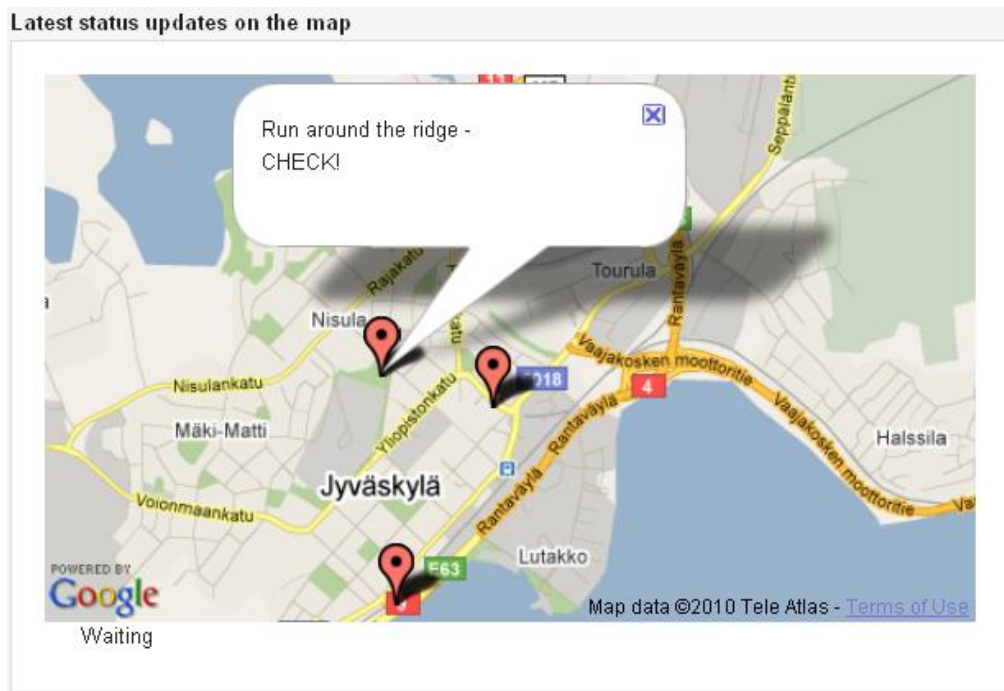


Figure 17 – Latest status updates on the map

## 5.5 Activity timeline

The activity timeline panel uses the Javascript widget from SIMILE (<http://www.simile-widgets.org/timeline/>) to visualize the stream of status updates on a timeline. the user can scroll the timeline back and forth and click on updates to display the whole message. The timeline contains new status updates from Facebook, network updates from LinkedIn and tweets from Twitter (see Figure 18).

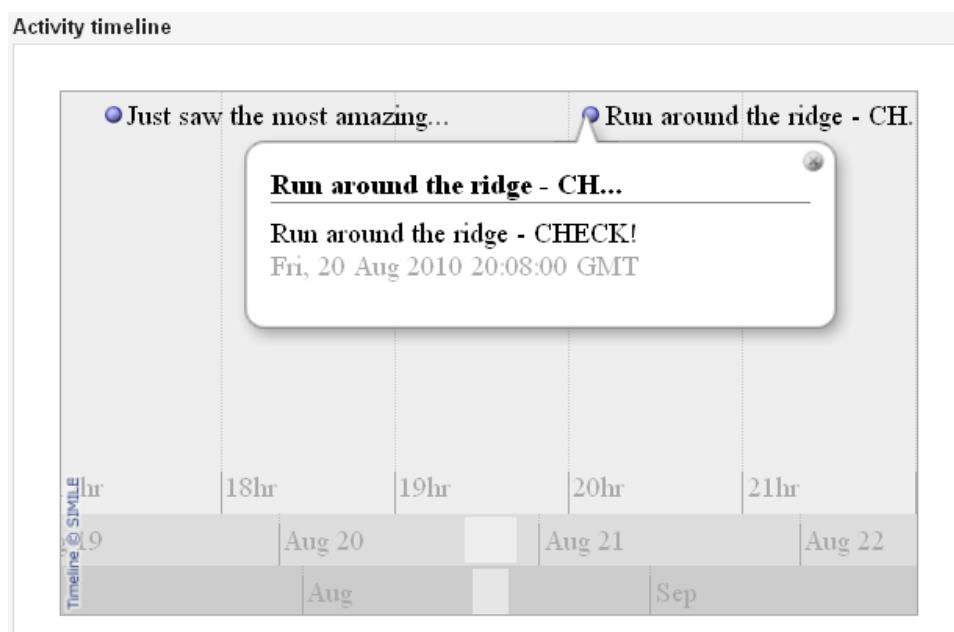


Figure 18 – Activity timeline

## 6 Meego version of Mashupper

The main functionality of the mobile version of the Mashupper is browsing the personal user network. The general idea of the user interface is similar to the web user interface. Nevertheless there are a few modifications due to the limited screen size of a mobile device. Furthermore, the UI is made fluid to make smooth touch handling of the browser possible.

After the user logs in using the same credentials as used in the web version of Mashupper, the user sees a list of his/her contacts (Figure 19).



Figure 19 – User contacts

After clicking (tapping) on a particular contact, the user sees profile pictures and links from all social networks that this friend can be found on (Figure 20). The link contained in the interface will open the browser on the mobile device pointing it to the web address where the complete profile can be viewed.



Figure 20 – List of friend's profiles on different social networks

The user can also choose to see friend's messages (tweets, status updates, etc.) in form of a list or on a map (Figure 21 and Figure 22).



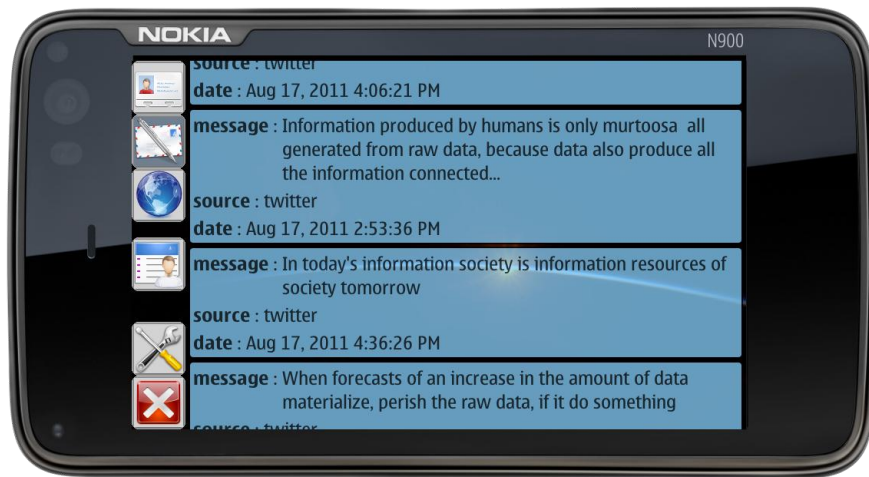


Figure 21 – List of friend's messages



Figure 22 – Friend's messages on a map

The functioning of the mobile interface can also be seen in the video delivered with this report.

## 7 Conclusions

By developing the Mashupper application we tried to demonstrate a new paradigm where the application is running both in the cloud and on the mobile phone. The cloud part of the application is based on the Ubiware platform. It performs CPU and network-intensive functions, such as gathering, updating, matching and integrating of social data from several sources. The mobile part of the application was developed using the Meego platform and it contains relatively light-weight interface that just displays the results provided by the cloud part of the application.